Date: 10/23/03	Express Mail Label No. EV215730718US

Inventors: Robert G. Noble, Douglas M. Laing, Andrew McBrien and Peter Ward

Attorney's Docket No.: 1086.2013-001

SYSTEM AND METHOD FOR ORGANIZING AND SHARING OF PROCESS PLANT DESIGN AND OPERATIONS DATA

RELATED APPLICATION

This application claims the benefit of U.S. Provisional Application No. 60/421,630, filed on October 25, 2002, the entire teachings of which are incorporated herein by reference.

BACKGROUND OF THE INVENTION

Process engineering involves the design and operation of a wide variety of processing plants and processes carried out therein. Such processes include, but are not limited to, chemical, petrochemical, refining, pharmaceutical, polymer, plastics and other process industries. In process engineering, a plethora of computer-based tools are employed by engineers to develop and evaluate new processes, design and retrofit plants, and optimize the operation of existing plants. For example, it is typical for several dozen separate engineering design tools to be used solely during the front-end engineering design phase of a new plant design.

Engineering tools tend to be developed to address a specific aspect of the entire process plant; as a result each different tool typically manages its internal engineering data using different definitions, formats, and schemas. As a result, efficient exchange of data between disparate engineering tools, although highly desirable, is very difficult in practice. This leads to handover inefficiencies, errors and rework, and lost opportunity to re-use engineering project knowledge, i.e. between the plant design and operations phases.

Within the past decade there has been growing recognition of the potential use of engineering database systems for the management of process data and the integration of disparate engineering tools. Typical implementations are developed by linking engineering tools to homegrown proprietary databases and data models using proprietary interfaces.

Similarly, once a plant is built, there is a need to implement computer programs that monitor and optimize plant operation or enable access to equipment, materials, instruments, operating procedures, maintenance orders, or other information about the plant. Until now, the process of implementing such computer programs for use in plant operations has been a manual entry of data and manual search for information stored in many systems.

This means that in the plant operation there is no use of the data that was generated during engineering design (or vice versa, no use of data generated during plant operation in engineering design). In addition, manual search for plant information through disparate computer systems creates a lot of inefficiencies and errors (e.g. wrong version of documents on specific topic is retrieved).

The growing experience of practitioners in this area has raised a number of practical problems:

Scope limitations: Data models developed to support specific engineering tools or engineering phases are limited to the narrow scope of those tools. For example, a data model developed to support process simulation tools does not have the scope to support plant pipe line-sizing tools; a data model developed to support plant equipment maintenance does not have the scope to support detailed mechanical equipment design. The use of such data models beyond their original intended scope of application is problematic.

Complexity: Attempts to support a wider scope of application will lead to more complex data models; for example, concepts of inheritance, parts, and hierarchy are needed to adequately model process equipment. The required complexity of the data models makes it practically infeasible for non-expert engineers to access useful data.

Data portability limitations: Proprietary data models developed from different sources will differ widely in formats, terminology and schemas. These wide differences make it practically infeasible to exchange data between systems that were developed independently, without prohibitively expensive efforts to map the two data models.

Maintainability: Engineering systems need to evolve over time, to accommodate new tools and uses. This will often require significant data model changes, which will in turn require re-mapping and re-linking of all of the engineering tools in the system. Maintenance of constantly evolving engineering systems is extremely costly.

Inconsistent data used in engineering design and in plant operations: Since an existing plant is constantly being modified, it is practically impossible through a manual data entry to keep an updated set of data to be used for engineering design.

As the plant matures, there is an accumulation of an ever increasing amount of operating procedures, knowledge about best operating conditions, performance, maintenance procedures, and other information. Efficient access to all information related to a particular equipment, instrument, material, process unit, entire plant, enterprise, etc. is essential for rapid information review to enable agile decision making.

Given the above, there is a need for improvement in the methodology and systems used for managing plant process and operations data.

SUMMARY OF THE INVENTION

The applicants have developed a system and method for organizing and sharing process plant design and operations data which:

--Is capable of supporting the full scope of process engineering activities in design and operations of the oil/gas/chemicals sectors and other manufacturing industries;

--Enables companies in these sectors to converge on a common data modeling system, thus enabling them to share data directly;

--Clearly separates data modeling and engineering tasks;

--Consolidates multiple engineering application representations of engineering data;

--Provides role-specific and application-specific views to the consolidated data model.

--Allows data model changes without upsetting applications of the data

During engineering design of a plant, data describing the plant (i.e., equipment, materials, operating conditions, feedstocks, products, instrumentation to measure behavior of the plant, plant topology, etc.) are defined. Once the plant is built, the same data describe the existing plant. The present invention enables data (describing the plant) that are defined during engineering design to be:

- Shared by all participants in the design work process, and
- Transferred from design to plant operation and be used as a basis for computer programs that enable retrieval of information about the plant, as well as to monitor and optimize plant operation.

Similarly, the invention described herein enables plant data (equipment, materials, operating conditions, feedstock, product, instrumentation, plant topology) that are created during implementation of computer programs that monitor and optimize plant operation, to be:

- Shared by all personnel in the plant, enterprise that owns the plant, or anyone authorized access to the data;
- Transferred from plant operations to engineering design (computer programs) thereby ensuring that the same data that describe operation and equipment of the existing plant are used in redesign of the plant.

In addition, the invention described herein enables all data (e.g. equipment size parameters) and information (e.g. operating procedure or maintenance order) about a particular part of the enterprise (e.g. a specific plant or a pipeline connecting the plant to a terminal), or part of the plant (e.g. heat exchanger), or part of the equipment (e.g. tube in a heat exchanger) to be:

- Viewed as it is most suitable for a particular job function, e.g. mechanical engineering will want to review size of the tubes in the heat exchangers, what material they are made of, etc., while a process engineer may want to view only the capacity of the heat exchanger.

- Accessed directly regardless where it is stored in various computer systems.

To accomplish this, the present invention in a preferred embodiment provides (1) a three-tier object-oriented data model architecture for organizing the data and for enabling access to information; (2) data schemas for objects and quantities typical to chemical plant process engineering and operations; and (3) a software system for the definition and administration of the data model, and management and access of data objects and data.

In particular, computer method and apparatus of the present invention for managing and sharing engineering data for chemical or other engineering processes include a respective class view for each of multiple software applications of interest, a composite class view, and a core (conceptual) data model. The class views, composite class view and core data model are consolidated or otherwise combined to form a multi-tier data model with links between corresponding attributes across the tiers. The multi-tier data model enables (i) management of engineering data from the multiple software applications, and (ii) sharing or access to information (engineering data) in the multiple software applications by other process and plant engineering routines and programs. An amalgamator synthesizes the class views, composite class view and conceptual data model into the multi-tier data model. In forming the multi-tier data model, there is a one-to-one mapping between an attribute in the class view and that of the composite class view, and a one-to-one mapping between an attribute in the composite class view and a data path in the core data model to corresponding software applications from which the attribute originated. Preferably, there are links from the composite class view attributes to the application class views.

In accordance with one aspect of the present invention, each class view is preferably represented in terms from the respective given application such that an end user of said given application is able to access data from the core data model.

Further, in the preferred embodiment, at least one of the class views, composite class views and core data model are represented by object oriented programming elements. Certain object oriented programming elements are defined by classes. The invention apparatus further comprises a class library editing subsystem for enabling user creation and editing of definitions of the classes.

Preferably, the class library editing subsystem employs XML for interfacing with users.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other objects, features and advantages of the invention will be apparent from the following more particular description of preferred embodiments of the invention, as illustrated in the accompanying drawings in which like reference characters refer to the same parts throughout the different views. The drawings are not necessarily to scale, emphasis instead being placed upon illustrating the principles of the invention.

Fig. 1 is a schematic diagram illustrating the three-tier data model architecture of the present invention.

Fig. 2 is a diagram describing the invention conceptual object classes in an embodiment for process unit operation models, process streams, plant equipment, piping systems, and instrumentation .

Figs. 3a - 3d are a set of tables illustrating a preferred embodiment of part of the conceptual data model (third tier) for a shell and tube heat exchanger equipment class used in the embodiment of Fig. 2.

Figs. 4a - 4k are a set of tables illustrating a preferred embodiment of the composite class view (second tier) for the shell and tube heat exchanger in the embodiment of Fig. 2.

Figs. 5a - 5b are a set of tables illustrating a preferred embodiment of two typical application class views (first tier) for shell and tube heat exchangers such as in the Fig. 2 embodiment.

Fig 6. is a block diagram of a data server system embodying the present invention.

Fig. 7 is a block diagram of a class library editor system of the present invention.

Fig. 8 is a flow diagram of the class library editor and data server systems of Figs. 6 and 7.

DETAILED DESCRIPTION OF THE INVENTION

A description of preferred embodiments of the invention follows.

As described herein, the present invention is intended to be used as a part of the software architecture in a computer software system for managing engineering data and integrating engineering tools used in the design and operation of plants in the process industries (chemical, petrochemical, refining, pharmaceutical, polymers, plastics and other process industries).

A. Three-tiered data model architecture

In one embodiment of the present invention, a data model architecture 10, as shown in the Fig. 1,is formed of three tiers:

(i) Class views 20 (Tier I);

(ii) Composite class views 30 (Tier II); and

(iii) Conceptual model 40 (Tier III).

A Class View 20 is an application view of the data, expressed in the language and terminology of the application 22 or application users. A Class View 20 comprises a formal representation of an individual application 22 data model. The terminology, structure and content of the Class View 20 follows closely the application's 22 own data representation. The development of Class Views 20 fulfills two major purposes. They 20 provide important documents as input to the generation of the conceptual model 40. They 20 also provide the primary tool for mapping an application 22 to the conceptual model 40, in a manner insulating applications 22 from changes in the underlying conceptual model 40.

The synthesis (or consolidation) of the Class Views 20 results in the creation of a Composite Class View 30. In particular, the composite Class View 30 is an amalgamation and rationalization of the individual Class Views 20. The descriptions of the attributes in the Class Views 20 remain in application domain terminology.

The Composite Class View 30 is subsequently mapped to a core or conceptual model 40. The conceptual model 40 defines the worldview of the data model, using abstraction and normalization to build a model that is consistent, compact, and maximizes reuse of conceptual constructs.

In the construction of the Composite Class View 30 and the Class Views 20, Applicants aim for a one-to-one mapping between an attribute in the Composite Class View 30 and a route in the conceptual model 40 (i.e., the route being a linked list or other data path of the corresponding applications 22 from which the attribute originated).

As illustrated in Fig. 1, engineering applications 22 are interfaced or integrated into the invention three-tier data model architecture 10. Engineering applications 22 include, but are not limited to, equipment or other data sheets, sizing calculations, instrument loop diagrams, plant asset database programs/systems, etc. As described above, there is a need and desire for these engineering applications 22 to be accessed by users of various roles. However, each application 22 displays the sought after data with different views.

In the invention three-tier data model architecture 10, class views 20 provide the application specific views at Tier I of the model 10. One or more composite class views 30 at Tier II provide consolidated engineering views. Each consolidated view ensures data is shared correctly by related applications 22. At Tier III, the invention conceptual model 40 provides a highly normalized, application-independent data model.

B. Description of a preferred embodiment of the three-tier data model

Fig. 2 gives a high-level view of the object classes for a preferred embodiment of a conceptual model 18 for process engineering. In particular, the process engineering conceptual model 18 includes representations of process unit operation models 24, plant equipment items 26, plant piping systems 21, and other plant equipment aggregate objects 28.

To further illustrate the model representation of equipment items 26, Figs. 3, 4 and 5 present a preferred embodiment of the three-tier data schema for the specific equipment type called "Shell and Tube Heat Exchanger" 32. Beginning with the conceptual model 18 (Tier I), Figs. 3a- 3d provide a representation of the structure and attributes part of the model 18 for a shell and tube heat exchanger equipment class 32. The illustrated shell and tube heat exchanger class 32 defines attributes by name 42, representation type 44 (i.e., real number, character string, table, etc.), quantity type 46 (units or format of measurement, e.g. flow rate mass, flow rate volume, flow rate moles, percentage, temperature, pressure, concentration mole/mole, etc.) and a description 48. This representation is only part of the conceptual data model 18.

Figs. 4a - 4k illustrate a table 34 of the attributes for a composite view 30 for a shell and tube heat exchanger 32. The "Route" column 36 in the table lists the attribute mappings between the conceptual data model 18 (Tier III) and the composite class view (Tier II) attributes 34.

Fig. 5a is a table 38 of some of the attributes for a specific application 22 class view 20 (Tier I), corresponding to that of a mechanical equipment data sheet view for a shell and tube exchanger 32. The "Link" column 39 in the table lists the attribute mappings between the composite class view (table 34 of Figs. 4a -4k) and the application class view attributes 38.

Fig. 5b is a table 37 of some of the attributes for an alternative application class view 20 (Tier I), corresponding to that of a thermal design calculation program 22 view for a shell and tube exchanger 32. The "Link" column 35 in the table lists the attribute mappings between the composite class view (table 34 of Figs. 4a - 4k) and this particular application class view attributes 37.

Arranging and relating the model 18 data in the foregoing manner according to the three-tier architecture 10 of the present invention provides certain advantages over the prior art as described above. These advantages will become clearer in the following discussion of management and use of the invention model representations.

Description of a class library editor system

In a preferred embodiment, the model representations (i.e., representations of the conceptual model 40 and various class views 30, 20) are implemented by objects in an object oriented programming language. Each object is defined by a class. Referring to Fig. 7, a Class Library Editor system 54 of the present invention supports the creation and editing of Class Libraries 52 and the compilation of these class libraries into a Class Store 50. The Class Store 50 is used in the data management system 60 (Fig. 6) of the present invention discussed later. A Class Library 52 comprises a named collection of related class and association definitions (organized according to an XML Schema). Typically, the set of definitions contained in each class library 52 are related by a common purpose, e.g. a class library 52 may contain classes for mechanical parts for rotating equipment, or classes of heat transfer equipment 26, 32.

The Class Store 50 is a 'compiled' version of the Class Library 52. It contains a 'flattened' (not hierarchical) representation of the classes in the Library 52 and its referenced libraries, with all dependencies checked and properly mapped.

The Class Library Editor system 54 supports multiple class libraries 52, and a class library may reference other class libraries to allow relationships between classes in different class libraries.

In the preferred embodiment of the Class Library Editor system 54 there is a class library editor 55. This editor 55 is formed of two subsystems, namely a user interface 56 and dynamic link library 58. The user interface 56

(i) Supports the interactive creation and editing of data model libraries 52.

(ii) Allows the data modeler to define and edit the class views 20, composite views 30 and classes and supporting data model constructs

(iii) Creates links between the attributes of an application class view 20 and the attributes in the composite view 30.

(iv) Creates routes between attributes in the composite view 30 and the attributes in the conceptual model 40.

(v) Checks the consistency of the definitions across the three tiers of the model 40.

(vi) Shows the usage of attributes in one tier by attributes in the tiers above.

(vii) Generates the Class Store 50 from multiple class libraries 52.

The dynamic link library 58 provides the underlying basic operations on the data model 40 such as insertion, deletion, renaming, and modifying the properties of data model constructs. The dynamic link library 58 also provides an automation interface to the data model 40 so that a data model can be manipulated programmatically as well as through the GUI 56.

Users can start developing the (Tier III) data model 40 at any of the three tiers I - III with views 20, 30, 40 (Fig. 1). For example, they can start by defining a particular application class view 20. They can then progress to linking the data representations from that view 20 to data representations in a new or existing composite view 30. Finally they can define new or reuse existing classes to describe the conceptual model 40.

Alternatively users can create an application independent composite view 30 in the second tier, and later create class views 20 which link to this composite view 30. The links between the three tiers in the data model 40 are preferably carried out (i.e., defined) by "drag and drop" operations in the user interface 56. Other user selection and/or command techniques are suitable.

D. Description of a data server system

Using the Class Store 50 generated by the Class Library editor 55, a data server system 60 instantiates and manages objects and their underlying data. The data server system 60 also manages access of the objects and data by external programs. A preferred embodiment of a data server system 60 of the present invention is shown in Fig. 6 and is described next.

The data server system 60 implements Object Models 12 (programming objects representing parts of given conceptual models 40 and corresponding composite class views 30 and application class views 20) including persisting and restoring them from a database management system 16 such as a RDB. The data server system 60 provides session management for applications 22 connecting to a workspace. The data server system 60 implements access control checks as determined by end user's roles. The data server 60 implements automation API (application program interface) for use by application data services and programming scripts.

In order to accomplish the foregoing, data server system 60 preferably includes a class store subsystem 59, an object store 62, a database query module 64, a database driver 66 and a transaction manager 68.

Subsystem Class Store 59

The class store subsystem 59 includes the C++ classes that provide the definitions for classes, attribute definitions and any other definition related information.

Subsystem Object Store 62

The object store sub-system 62 covers the classes required to implement objects 12, attributes, cases and any other instance related information.

Subsystem DBQuery 64

The query module (DBQuery) 64 defines the automation API's to other external systems. It is through these API's that both application data services and KB scripts can interface to the other subsystems 59, 62, 64, 66, and 68.

Subsystem RDB Driver 66

This subsystem 66 manages the communication between the Object Store 62 and the relational database(s)16. In one embodiment, a generic OLEDB implementation is employed. However, the system is architected such that drivers for specific types of relational databases may be incorporated (e.g. a dedicated Oracle driver using native Oracle API's).

Subsystem Transaction Management 68

The transaction manager/subsystem 68 addresses transaction management issues of the system 60. This includes maintaining a request queue and a thread pool that reads from the request queue. The transaction manager 68 also optimizes transaction locking and detects transaction deadlocking using various known techniques.

The combination of the class library editor system 54 and the data server system 60 thus provides a method of creating and using data models as illustrated in Fig. 8. Given are one or more software applications that model processes of interest (e.g., chemical engineering processes). Each software application has or follows a respective data model. For each such application, the class library editor system 54 provides (step 100) a practitioner's (end user of the application) view of the application using a class view 20 of the application data model.

Next (step 102) the class library editor system 54 consolidates the class views 20 into a composite class view 30.

Next (step 103) the class library editor 55 generates new or edits existing class structures and attributes of classes comprising the conceptual data model 40. The class library editor system 54 then creates a class store 50 (step 104) for the three tier data model. The class store 50 comprises class attributes and structures of class views 20, composite class views 30 and core conceptual model 40 with accompanying attribute links between the tiers.

A class store 50 results from step 104 having been accomplished. In turn (step 106), data server system 60 uses class store 50 to instantiate data objects 12 and expose these objects 12 through interfaces following the class and other views 20, 30, 40. The combined systems 54, 60 enable sharing of original application data (e.g., engineering data) with other process and plant engineering routines, programs and the like. As such the invention enhances process engineering as heretofore unachieved by the prior art.

While this invention has been particularly shown and described with references to preferred embodiments thereof, it will be understood by those skilled in the art that various changes in form and details may be made therein without departing from the scope of the invention encompassed by the appended claims.

For example, the use of chemical engineering processes is by way of illustration and not limitation of the data modeling and management applications to which the present invention is directed. Modeling of refining, pharmaceutical, polymer processing and other engineering processing are contemplated/included.

Amalgamation and rationalization at the composite class view 30 stage (Tier II) of the present invention is by known techniques. Similarly, abstraction and normalization at the conceptual model 40 (Tier III) is by known techniques. A variety of techniques or combinations thereof are suitable in forming the composite class view 30 and conceptual/core model 40, given the foregoing description of the invention and preferred embodiments.

Further, one or more databases 16 on one or a network of computer systems may be employed. Data server system 60 may be software or a mixture of hardware and software executed on a digital processor or suitable computer system (stand alone, local area networked, wide area networked and the like). Various computer configurations are suitable and within the purview of one skilled in the art.

Although the preferred embodiment is described with three tiers, it is understood that other multiple numbers of tiers are within the purview of one skilled in the art given the above description of the present invention. The effect of the described three tiers may be implemented by any number of tiers or one partitioned tier and the like.